Projet : TIC-TAC-TOE AVEC PYTHON

Réalisé par : MAHAMAT IBRAHIM ISSA GUIRE

YASSER

PLAN DU TRAVAIL :

1. Introduction
2. Reinforcement Learning
3. Conception du jeu
4. Création de la table Q et de l’agent
5. Conclusion
6. Introduction :

Ce projet à pour but d’apprendre et de mettre en application les notions des cours d’apprentissage par renforcement (Reinforcement Learning en anglais (RL)). Pour ce faire nous allons tout d’abord définir le RL. Et ensuite prendre l’exemple du jeu TIC-TAC-TOE tout en expliquant la règle de notre jeu ainsi que les étapes à suivre pour la mise en application de ce dernier.

1. Reinforcement Learning (RL) :

L’apprentissage par renforcement est une branche de l’intelligence artificielle. C’est la science de la prise de décisions optimales à l'aide d'expériences.

L’objectif est de maximiser la récompense d'un agent en prenant une série d'actions en réponse à un environnement dynamique. Elle se décompose en étapes suivantes :

* Observation de l'environnement
* Décider comment agir en utilisant une certaine stratégie
* Agir en conséquence
* Recevoir une récompense ou une pénalité
* Tirer les leçons des expériences et affiner notre stratégie
* Itérer jusqu'à ce qu'une stratégie optimale soit trouvée

Il existe 2 principaux types d'algorithmes RL. Ils sont basés sur des modèles et sans modèle. Un algorithme sans modèle est un algorithme qui estime la politique optimale sans utiliser ni estimer la dynamique (fonctions de transition et de récompense) de l'environnement. Alors qu'un algorithme basé sur un modèle est un algorithme qui utilise la fonction de transition (et la fonction de récompense) afin d'estimer la politique optimale.

1. Conception du jeu :

Le tic-tac-toe appelé aussi morpion est un jeu basic et simple sous forme d’un tableau contenant 9 cases, qui se joue à deux (avoir deux joueurs). Les principes de ce jeu sont les suivantes :

* Chaque joueur choisira un caractère (soit X ou O) ;
* L’un des deux joueurs commencera le jeu cochant soit X ou O sur une case vide de son choix et le second joueur le suivra à son tour en cochant aussi une case vide ;
* Le premier joueur arrivant à aligner les trois caractères de son choix (X ou O) en diagonale, vertical ou horizontal est déclarer vainqueur de la partie (voir figure 1 ci-dessous) ;
* Si le tableau est rempli par les caractères (X et O) sans alignement en vertical, horizontal ou diagonal alors les deux joueurs sont à égalité par conséquent match nul ;

Figure 1 :

Une image contenant shoji, mots croisés, bâtiment, horloge

Description générée automatiquement

Pour ce faire nous avons créé une classe « TicTacToe ». Cette classe est composée des plusieurs fonctions à savoir :

* La fonction « play » accepte x et y comme arguments et place le mouvement du joueur actuel à ces coordonnées sur le plateau 3x3 du jeu, puis change le tour de l'autre joueur. Si nous avons un gagnant, il retournera le gagnant sinon il reviendra None.
* La fonction « get\_winner » pour détecter le gagnant ;
* La « is\_ended » fonction vérifie s'il reste des cellules vides dans le plateau et renvoie si le jeu est terminé ou non ;
* La « get\_valid\_actions » fonction renverra toutes les coordonnées des cellules vides. Il sera utile pour former notre agent d'IA.

La structure du jeu est donc la suivante : nous devrions appeler la fonction « play » en boucle jusqu'à ce que nous ayons un gagnant ou que le jeu se termine par un match nul.

1. Création de la table Q et de l’agent :

À chaque tour, notre agent IA doit décider quelle cellule choisir pour placer son mouvement. Pour ce faire, il doit avoir une estimation des récompenses pour chaque (état, action) paire afin de pouvoir choisir l'action qui entraîne la récompense maximale. La table Q contiendra ces estimations telles que :

Q[état][action] 🡪 estimation de récompense.

Maintenant pour estimer les « récompense », nous utiliserons l'algorithme Q-learning pour mettre à jour ces valeurs de manière itérative jusqu'à ce qu'elles convergent vers les valeurs réelles.

1. Le processus d'apprentissage :

L’apprentissage de l’agent se fait de la manière suivante :

* L'agent obtient l'état actuel du jeu et choisit une action. Au début, les valeurs Q ne sont pas mises à jour et nous voulons que l'agent explore le jeu. Nous y ajoutons donc un peu d'aléatoire. Il choisira une action aléatoire avec une probabilité de pou choisira l'action avec une récompense estimée maximale avec une probabilité de 1-p. Au fur et à mesure que l'entraînement se poursuit et que l'agent explore l'environnement de jeu, nous diminuons progressivement p de 1 à 0 afin qu'à la fin l'agent ne choisisse que la meilleure action.
* L'agent applique l'action choisie au jeu. Ensuite, en tant qu'adversaire, nous jouons un coup aléatoire. Nous avons maintenant le nouvel état. Si l'agent gagne, nous avons une récompense de 100. S'il perd, nous utilisons -100 comme récompense. Sinon la récompense est de 0.
* Nous pouvons maintenant mettre à jour la valeur Q de la paire (état, action) avec cette formule :

Q (S, A) = Q (S, A) + α ∗ (R + Γ ∗ maxi (S′, a) – Q (S, A)): equation de Bellman.

S est l'état actuel et R est la récompense instantanée que nous avons obtenue après avoir appliqué l'action. A est l'action choisie, αest la taille du pas qui signifie combien de poids nous voulons donner à la nouvelle valeur par rapport à la valeur actuelle. Γ est le facteur de remise. Cela signifie le poids que nous accordons aux récompenses futures par rapport à la récompense instantanée. Maxi (S′, a) est la valeur Q maximale sur toutes les actions pour le nouvel état.

* Répétez les 3 étapes ci-dessus jusqu'à ce que les estimations de Q soient suffisamment bonnes. Si le jeu se termine, créez-en un nouveau et continuez la boucle.

Conclusion :

En somme à l’issue de ce projet nous avons eu l’occasion de programmer le jeu morpion (tic-tac-toe) avec IA sans faire appel à aucune bibliothèque externe. Nous avons aussi eu aussi l’occasion d’apprendre davantage sur l’algorithme Q-learning.

En effet nous avons appris que le Q-learning est un algorithme d'apprentissage par renforcement sans modèle. C’est la valeur attendue (récompense actualisée cumulative) de faire une action a dans l'état s puis de suivre la politique optimale. Il faut aussi noter qu’il existe d’autre façon de programmer le jeu morpion. En utilisant par exemple l’algorithme du minimax ou bien la librairie Pygame.